

## SYSTEM AND METHOD FOR MULTI-SAMPLING PRIMITIVES TO REDUCE ALIASING

### TECHNICAL FIELD

The present invention is related generally to the field of sampled imagery,  
5 and more particularly, to systems and methods for generating antialiased, sampled images.

### BACKGROUND OF THE INVENTION

In computer graphics, a sampled image is created from input geometry or  
mathematical computations that typically represent a scene. The sampled two-dimensional  
(2D) image is represented by an array of discrete units referred to as pixels. The pixels are  
10 generally arranged along two perpendicular axes corresponding to the horizontal and  
vertical axes of the image. The color or monochrome values of the pixels are determined  
by calculated sample values, typically derived from scene data, lighting data, or other input  
data. The image represented by the array of pixels is typically transferred to a visual  
medium, such as by being printed onto paper or film, or displayed upon a computer display  
15 device. The number of pixels and number of colors or values used in rendering a graphics  
image limit, to some extent, the visual qualities and characteristics of the viewed image, for  
example, the image definition, smoothness, and detail. Consequently, a great amount of  
effort has been devoted to developing sophisticated graphics processing and rendering  
techniques for higher image quality.

20 In addition to high-quality static images, high-quality animation depicting  
visual effects of motion, is also desirable. The illusion of motion is created by quickly  
displaying related images in a sequence of image frames in which an object appears in  
different positions. Although the motion of the object may appear to be continuous, each  
frame of the animation is a separate image that is displayed momentarily. Thus, the quality  
25 of each frame will affect the quality of the animation or the illusion of motion. Depending  
upon the rendering algorithm, representing motion may result in the creation of motion

artifacts. The more noticeable effects include static edge-effects such as "staircasing," as well as "crawling" and "popping" of pixels on an edge of a moving object having a value that contrasts against a background having a different value.

For example, consider an object having a straight edge and of a first color  
5 which appears to be moving across a background of a second color. As the edge of the object moves across a pixel, there must be a determination as to when the color of the pixel changes from the color of the background to the color of the object. If a single sample location within the pixel is taken to determine its color, then, when the edge of the object passes the single sampling location, the color of the pixel is changed. The location of a  
10 single sample location is typically located at the center of the pixel, and thus, the value of the pixel is determined by the value calculated for the center of the pixel. As a result of a single sample point determination, the edge of an object may extend well into the region represented by the pixel, but because the edge has not reached the center (*i.e.*, the sample point) of the pixel, the pixel may continue to have the color of the background. As a result,  
15 when multiple images or frames are displayed in sequence to provide the illusion of objects in motion, pixels along the edge of an object may "pop" from one color or value to another. This effect can be distracting for a viewer. The relative motion of an object with respect to the orientation of the pixels of the image may be such that pixels along the edge of the object pop values in a manner and with a regularity that creates a visual effect of the edge  
20 of the object "crawling."

There have been many different approaches to addressing the issue of aliasing. One such approach is to increase the resolution, or the number of pixels used to represent an image. Although the available resolution of computer graphics displays and computer printing devices have reduced aliasing, there are practical limitations on the  
25 manufacture of displays and other devices used for forming or recording an image which will limit the maximum available resolution. Moreover, no matter how high a (finite) resolution is used to represent a graphics image, so long as the value of each pixel is only approximated from calculated values for discrete, sample points, within the pixel, the

image is subject to some degree of aliasing, and a sequence of such images is subject to motion artifacts.

In another approach, aliasing can be "antialiased" to some degree by calculating multiple number of values in determining a final value for each pixel. That is, multi-sampling systems for antialiased rendering use sample values from multiple samples taken from a pixel region in determining the value of the respective pixel. For example, when multi-sampling is applied to the previous example, as the edge of the object passes the first of the sample locations in a pixel region, the pixel is given a value that is a compromise between the value of the object and the disparate value of the background. Where the size, shape and motion of the object result in the object gradually and completely covering the pixel, the value of the pixel will change each time the object covers another sample location until all of the sample points share the color of the object, at which time the pixel takes on the color of the object.

One approach to implementing the previously described antialiasing technique has been to shift the input geometry of the entire three-dimensional (3D) environment a sub-pixel distance for each of the sub-pixel sample locations and then repeat the entire rendering or rasterization process. In effect, the rendering process, from generating scene geometry through rasterizing the scene, is repeated for the calculation of each of the multiple samples. Although the resulting image is antialiased, this approach consumes considerable processing resources of a graphics processing system because multiple passes through the process of generating the scene geometry are needed. The time necessary to generate the multiple frames is a linear function of the number of desired samples, and as a result, no economy of scale performance gains would typically be realized. Moreover, the generating the geometry of the 3D environment multiple times necessitates either application driven generation of the frames, or an arbitrary amount of buffering within the graphics processing system to store the resulting data for an arbitrarily complex scene.

Therefore, there is a need for an alternative system and method for rendering an image from a representation of a scene, while reducing aliasing, such as motion artifacts and edge effects.

## SUMMARY OF THE INVENTION

5           The present invention relates to a method and system for performing multi-sample, antialiased rendering of images by performing multi-sample antialiasing at the primitive level. Values for pixels of an image of an environment represented by geometric primitives are calculated by the system. Geometric data is used to represent the geometric primitives, which are set-up into a scene of the environment. A geometric primitive of the  
10 scene is shifted by a sub-pixel offset and rendered to generate values for pixels of an intermediate image. The shifting and rendering is repeated for the geometric primitive, each time with the values generated at the respective location stored to a buffer. The stored values for the pixels of the intermediate images are combined to produce values for the respective pixels of the resulting image.

## 15 BRIEF DESCRIPTION OF THE DRAWINGS

Figure 1 is a flow diagram for conventional graphics pipelined processing.

Figures 2a and 2b are a flow diagram for graphics processing according to aspects of the present invention.

Figure 3 is a diagram of a multi-sample pattern according to another  
20 embodiment of the present invention.

Figure 4 is a block diagram of a graphics processing system according to an embodiment of the present invention.

Figure 5 is a block diagram of a computer system including the graphics processing system of Figure 4.

## DETAILED DESCRIPTION OF THE INVENTION

Embodiments of the present invention render antialiased images by reducing artifacts in both static images and also in motion pictures or animation. The systems and methods described herein perform antialiased rendering through a process of taking multiple samples, or "over-sampling" at the primitive or primitive set level for calculating the colors or values of pixels forming a resulting computer graphics image. Certain details are set forth below to provide a sufficient understanding of the invention. However, it will be clear to one skilled in the art that the invention may be practiced without these particular details. In other instances, well-known circuits, control signals, timing protocols, and software operations have not been shown in detail in order to avoid unnecessarily obscuring the invention.

Illustrated in Figure 1 is a process flow diagram 100 of a conventional 3D graphics pipeline process. It will be appreciated that Figure 1 does not necessarily represent a physical model of a graphics pipeline since the stages illustrated can be implemented in either software or hardware. Graphics information stored in a database is provided to the graphics pipeline (step 104) to begin construction of the 3D environment. The type of geometric data typically included in the database define vertices of geometric primitives having surfaces that are used construct the 3D objects, as well as attributes of the respective primitives, such as surface color, normal vector information, specular value, and the like. Along with the geometric data, environmental information is also stored. Environmental information includes such data as the number, color, and location of illumination sources, atmospheric properties, and the like.

The geometric data is typically provided to the graphics pipeline process in an object coordinate space and requires transformation into a world coordinate space (step 108). Various conventional transformation techniques are well known, and consequently, will not be discussed in detail herein. A viewpoint from which the 3D environment will be viewed is determined and a corresponding view volume is constructed. In order to ease the processing burden, the primitives located outside of the view volume are rejected, and the

primitives that are partly inside the view volume are clipped (step 112). Thus, only the primitives contributing to the resulting image need to be processed through the graphics pipeline.

Following the primitive clipping process, the vertices of the of the  
5 primitives to be processed are transformed from the world space to a screen coordinate space corresponding to the 2D representation of the 3D environment. The attributes for the primitives, such as for the surface normal vector for the surface of a primitive, are transformed to the screen coordinate space as well (step 116). Additional steps to setup the primitives take place after the transformation to the screen space (step 120), such as texture  
10 application. A rasterization or rendering step is then performed which generates pixel attributes for each primitive (step 124). The pixel attribute data are typically written to a pixel buffer or frame buffer. The pixel attribute data stored in the buffer are then used for constructing a resulting 2D image of the scene. Although embodiments of the present invention will be described herein as being applied to 3D graphics, it will be appreciated  
15 that the present invention may be applied in performing antialiasing to 2D graphics as well.

As mentioned previously, conventional approaches to antialiasing have been to either shift the geometry of the 3D environment a sub-pixel distance for each sub-pixel sample locations, and repeat the entire graphics pipeline process for each of the samples, or render a frame multiple times with the pixel values written into separate buffers. For  
20 example, where four sub-pixel samples are calculated for the attributes of a pixel, several of steps 104-124 of Figure 1 are repeated four times, each time with an entire frame shifted to a different sub-pixel position. The pixel attribute data resulting from each pass through the graphics pipeline process are written to a different pixel buffer, and then subsequently combined to determine final attributes of pixels for the final image of the scene. However,  
25 as also mentioned previously, the technique of rendering a frame multiple times consumes considerable processing resources of a graphics processing system because it requires four passes through the process of generating the scene geometry. The time necessary to generate the multiple frames is a linear function of the number of desired samples, and as a

result, no economy of scale performance gains would typically be realized. Moreover, the generating the geometry of the 3D environment multiple times necessitates either application driven generation of the frames, or an arbitrary amount of buffering within the graphics processing system to store the resulting data for an arbitrarily complex scene.

5 In contrast to the conventional approach, embodiments of the present invention reduce the impact a multi-sample antialiasing operation has on processing overhead by performing multi-sampling at the primitive level. That is, rather than rendering an entire frame multiple times, it is the geometric primitives themselves that are rendered multiple times. As a result, only the drawing operation for each of the primitives  
10 is repeated for each of the sub-pixel samples, rather than repeating multiple passes through the process of generating the geometry for the multiple scenes. Additionally, cache mechanisms already operating within a graphics pipeline, for example, texture caching, permits rapid rendering of the multiple instances of the primitive.

15 Illustrated in Figures 2a and 2b are a process flow diagram 200 of an embodiment of the present invention. It will be appreciated that the process flow 200 of Figures 2a and 2b is provided as an example to illustrate aspects of the present invention. Consequently, particular steps as described below may be removed or reorganized without departing from the scope of the present invention.

Steps 204-220 of the process flow are conventional and are similar to steps  
20 104-120 of Figure 1. However, at step 224, the primitive that is to be rasterized is shifted by a sub-pixel amount prior to rasterization (step 224). The primitive is shifted to a position corresponding to the location of one of the sub-pixel samples and then rasterized and the resulting sample values are written to a pixel buffer (step 228). The process of shifting the primitive, rasterizing and writing the resulting sample values to a buffer is  
25 repeated for each of the sub-pixel sample locations or until the appropriate number of samples are taken for a pixel (steps 232-240). The rasterization or rendering of the shifted primitive can be typically implemented using conventional circuitry and processing commands. For example, the triangle engine could be instructed to draw each primitive

multiple times by reissuing a draw command, each time shifting one of the vertices of the primitive a sub-pixel offset plus a memory buffer offset, and then re-rendering the primitive. The shifting and rendering process is performed at the primitive level until all of the primitives of the scene are rendered (steps 244-246).

5           The sample values in the respective buffers are then combined to produce final values for pixels of the image (step 248). Many well-known conventional blending or combining techniques can be used for the present invention. In one embodiment, the sample values from each of the different buffers for a respective pixel are averaged together. That is, each of the sample values contribute equally to the final value of the  
10 pixel, such as in a simple box filter. In another embodiment, the sample values are weighted as a function of the respective offset distances from the center of the pixel. For example, sample values calculated at sub-pixel sample locations located at greater distances from the pixel center will contribute less to the final value of the pixel than those calculated at sample locations closer to the pixel center. For example, a Gaussian weighting function  
15 can be used.

          In an alternative embodiment, sets of primitives, such as a tristrip, are re-rendered instead of an individual primitive. It will be appreciated that other known collections of geometric primitives can be used as well without departing from the scope of the present invention, such as triangle fans, sets of disjoint triangles, and the like. In the  
20 embodiment where the set of primitives is a tristrip, the tristrip definition should be buffered so that it may be reissued multiple times. However, it may be the case that the buffers of the graphics pipeline is insufficient to buffer enough geometric state for more than only a few triangles. Thus, consideration of the available pipeline resources should be considered when implementing this embodiment.

25           In alternative embodiments of the present invention, the use of rendering buffers can be economized. For example, with some loss of sampling accuracy,  $n$  instances of pixel color buffers and one z-buffer could be utilized. That is, for the  $n-1$  issues of the primitive, writing to the z-buffer is inhibited. However, the  $n$ th issue of the primitive is



rendered with the z-buffer inhibit removed. Thus, only one z-buffer is required for implementing the present invention. It will be appreciated that embodiments of the present invention may be implemented using other buffers whose state affects the rendering process, and which can be modified as part of the rendering process.

5           Figure 3 illustrates an example of a sampling pattern 300 that can be used in embodiments of the present invention. A pixel region 300 is shown with sampling locations 304 and 308. The pixel region 300 represents the region that a pixel of the resulting image covers. As shown in Figure 3, the pixel region 300 is dividend into four equal sub-regions. The sampling locations 304 and 308 are positioned at the center of the  
10 upper-left sub-region and the lower-right sub-region. Note that the sampling locations 304 and 308 are offset from a center 312 of the pixel region 300 by sub-pixel offsets  $d_1$  and  $d_2$ , respectively. Shifting a geometric primitive by the offset  $d_1$  will effectively result in a sample being calculated at the sampling location 304. Similarly, shifting the geometric primitive by the offset  $d_2$  will effectively result in a sample being calculated at the sampling  
15 location 308. Thus, by shifting the geometric primitive by the respective offsets, multiple samples for a pixel may be calculated and then used for antialiasing.

The sampling pattern 300 may be repeated for each of the pixels during the rendering of the primitive by selecting offsets that shift the primitive to a location that corresponds to the sampling locations of the sampling pattern 300. Alternatively, the  
20 sampling pattern 300 can be used as a basis from which dynamic offsets could be determined stochastically to implement "jittered" antialiasing. Both embodiments can be implemented using well-known conventional techniques. Further examples of the different types of sampling patterns and techniques are described in more detail in co-pending U.S. Patent Application No. 09/823,935, entitled MULTI-SAMPLE METHOD AND SYSTEM  
25 FOR RENDERING ANTIALIASED IMAGES, to Peterson *et al.*, filed March 30, 2001, which is incorporated herein by reference. It will be appreciated, however, that many different sampling patterns could be used in embodiments of the present invention in addition to those described herein and in the aforementioned co-pending patent application.

Therefore, the scope of the present invention should not be limited by the particular sampling pattern employed, except to the extent the limitation is recited in one of the claims attached hereto.

Figure 4 illustrates circuitry included within the graphics processing system 400 for performing various graphics and video functions, including antialiasing according to embodiments of the present invention. In the following description, the various stages of the graphics processing system 400 will be described as performing various functions and graphics processes. However, the various functions and graphics processes may also be performed by or associated with a different stage than described herein. Consequently, it will be appreciated that the particular embodiment of the graphics processing system 400 is provided merely as an example, and is not intended to limit the scope of the present invention.

As shown in Figure 4, a bus interface 402 couples the graphics processing system 400 to the expansion bus 434 or optionally a high-speed bus 436. In the case where the graphics processing system 400 is coupled through the high-speed data bus 436, the bus interface 402 will include a DMA controller (not shown) to coordinate transfer of data to and from the graphics processing system 400. A graphics processor 404 is coupled to the bus interface 402 and is designed to perform various graphics and video processing functions, such as, but not limited to, generating vertex data and performing vertex transformations for polygon graphics primitives that are used to model 3D objects. The graphics processor 404 is coupled to a triangle engine 408 that includes circuitry for performing various graphics functions, such as attribute transformations, rendering of graphics primitives, and generating texture coordinates for a texture map. The circuitry of the triangle engine 408 can also be used to perform antialiasing in accordance with embodiments of the present invention by over-sampling within the pixels of the graphics image to reduce aliasing artifacts.

A pixel engine 412 is coupled to receive the graphics data generated by the triangle engine 408. The pixel engine 412 contains circuitry for performing various

graphics functions, such as, but not limited to, texture application or mapping, bilinear filtering, fog, blending, and color space conversion. A memory controller 416 coupled to the pixel engine 412 and the graphics processor 404 handles memory requests to and from a local memory 420. The local memory 420 stores graphics data, such as pixel values. A display controller 424 is coupled to the memory controller 416 to receive processed values for pixels that are to be displayed. The output values from the display controller 424 are subsequently provided to a display driver 432 that includes circuitry to provide digital signals, or convert digital signals to analog signals, to drive a display device (not shown).

It will be appreciated that additional processes and circuitry that have not been shown or described with respect to Figure 4 may also be included in the graphics processing system 400. For example, various data buffers used throughout the graphics processing operation, such as a pixel buffer, a z-buffer, and stencil buffer, are included in the different functional blocks of the graphics processing system 400. However, they have not been discussed in detail in order to avoid unnecessarily obscuring the invention. It will be further appreciated that the circuitry included in the functional blocks of the graphics processing system 400 are conventional in design and are well understood by those of ordinary skill in the art. Consequently, the description provided herein is sufficient to enable the same to practice embodiments of the present invention.

As previously mentioned, the graphics processing system 400 facilitates antialiasing by over-sampling for at least some pixels of a graphics image. The sample values for the multiple samples for a pixel are calculated for each triangle, or other primitive, covering, or partially covering, the pixel. Calculation of the multiple sample values occurs at the primitive or primitive set level to avoid repeating many of the processing intensive steps of generating the geometry of the 3D environment. Each primitive is shifted by a sub-pixel offset and rasterized for the multiple samples. The sample value for a given sample position within a given pixel may be then stored or altered, for example, by well-known z-buffer techniques, in accordance with the processing of other data, such as other primitives touching the same sample position in the same pixel. In one

embodiment, the final sample values for a given pixel are subsequently combined in the pixel engine 412 or texture engine (not shown) or blend unit (not shown), to determine a sample value for the pixel in the resulting image.

Although the antialiasing operation has been described as being  
5 implemented in the triangle engine 408, it will be appreciated that the antialiasing operation may also be implemented in circuitry that is separate from the triangle engine 408, or included in one of the aforementioned circuit blocks of the graphics processing system 400, such as the pixel engine 412. Alternatively, the antialiasing operation may be divided and carried out by various circuit blocks of the graphics processing system 400. For example,  
10 portions of the antialiasing operation may be shared between the triangle engine 408 and the pixel engine 412. That is, when geometric primitives are rendered by the triangle engine 408, sample values are calculated for each sample point, typically of each pixel having a portion covered by the geometric primitive. Once the over-sampling is performed, the pixels may be further processed, for example, for depth, texture, and the like, and then  
15 the pixel engine 412 may perform the down-sampling or combining of the multiple sample values. Therefore, the particular implementation of the antialiasing circuitry 414 is a detail that may be modified without deviating from the subject matter of the invention, and should not be used in limiting the scope of the present invention. Moreover, embodiments of the present invention may be implemented in software or hardware, or a combination thereof. Those of ordinary skill in the art have sufficient understanding to implement the  
20 present invention as such.

It will be appreciated that the sample values discussed herein may represent values such as color values, grayscale values, depth values, and the like, at a sample location. It will be further appreciated that the sample values for the samples may consist  
25 of several different components. For example, a sample value may represent not only a pixel color, as a combination of red, green, and blue color components, but also a depth value. Another example includes sample values representing pixel colors which are combinations of luma and chroma components. Consequently, because it is well

understood in the art, although circuitry to perform graphics operation for each of the components is not expressly shown or described herein, embodiments of the present invention include circuitry, control signals, and the like necessary to perform operations on each component for multi-component sample values.

5               Figure 5 illustrates a computer system 500 in which embodiments of the present invention are implemented. The computer system 500 includes a processor 504 coupled to a memory 508 through a memory/bus interface 512. The memory/bus interface 512 is coupled to an expansion bus 516, such as an industry standard architecture (ISA) bus or a peripheral component interconnect (PCI) bus. The computer system 500 also includes  
10 one or more input devices 520, such as a keypad or a mouse, coupled to the processor 504 through the expansion bus 516 and the memory/bus interface 512. The input devices 520 allow an operator or an electronic device to input data to the computer system 500. One or more output devices 524 are coupled to the processor 504 to receive output data generated by the processor 504. The output devices 524 are coupled to the processor 504 through the  
15 expansion bus 516 and memory/bus interface 512. Examples of output devices 524 include printers and a sound card driving audio speakers. One or more data storage devices 528 are coupled to the processor 504 through the memory/bus interface 512 and the expansion bus 516 to store data in, or retrieve data from, storage media (not shown). Examples of storage devices 528 and storage media include fixed disk drives, floppy disk drives, tape cassettes  
20 and compact-disc read-only memory drives.

The computer system 500 further includes the graphics processing system 400 (Figure 4) coupled to the processor 504 through the expansion bus 516 and memory/bus interface 512. Optionally, the graphics processing system 400 may be coupled to the processor 504 and the memory 508 through other types of architectures. For  
25 example, the graphics processing system 400 may be coupled through the memory/bus interface 512 and a high speed bus 536, such as an accelerated graphics port (AGP), to provide the graphics processing system 400 with direct memory access (DMA) to the memory 508. That is, the high speed bus 536 and memory bus interface 512 allow the

5

10